ML Process

It might be difficult to have a generalized process. It seems there are a few distinguishing factors:

* Type of problem: supervised vs. unsupervised
  + This will determine if there are targets or only features
  + Maybe you might still want to do some data unsupervised analysis or exploration even if you have supervised data
  + For real world problems, especially in Kaggle, you will not have labels for testing set
* Size of input features
  + Some problems are intrinsically high feature: image recognition, etc., certainly no need for direct visualization; possible visualization after PCA, ICA, etc.
  + Some problems might be somewhere in between, e.g. Expedia (not sure how to deal with this yet)
* Data types: this greatly affects visualization and pre-processing
  + Inputs
    - Continuous (numerical): no problem
    - Categorical
      * Binary: ok, although that feature space is probably large and it is more of a different type of problem
      * Multiple values: tricky and even trickier if you convert it into dummies because it greatly expands the feature space
    - Text
      * Potentially very unstructured and might need to be dealt with on a case by case basis
    - Worst of all is when you have a combination of data types
      * In this case it is probably best to convert data into buckets
  + Outputs
    - Type: continuous, digital or multi-class
    - Class or probability
      * I haven’t looked into outputting probabilities

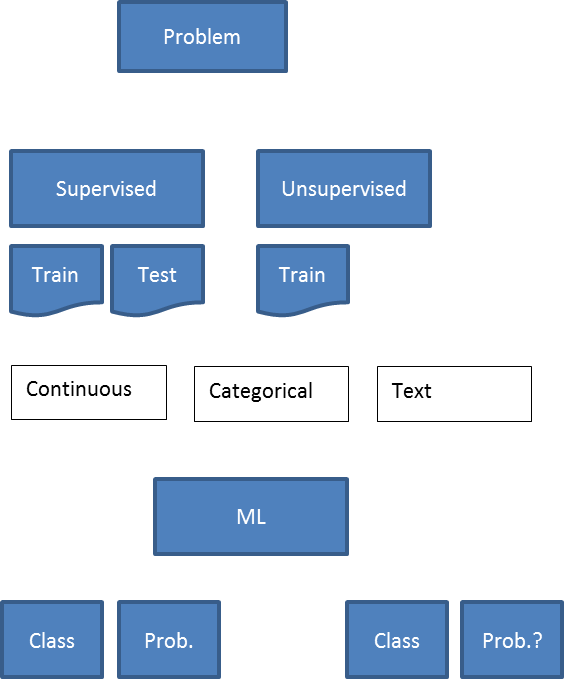
In addition there are 3 steps:

* Reshuffling
* Splitting training and testing sets
* Cross-validation

And importantly there is the first data exploration phase aimed at gaining some intuition on the data:

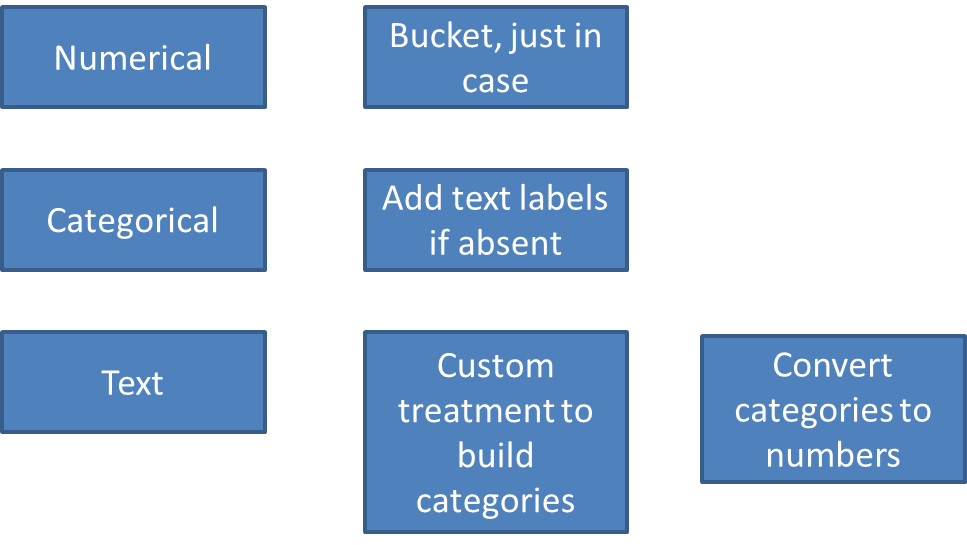
* Gathering stats on the data
* Plotting

All these are related to the type of data that you need to handle.

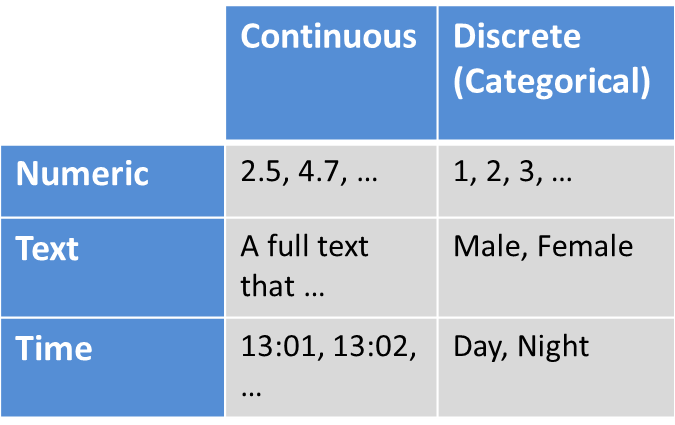


Maybe we start with:

1. Load data (different process for supervised / unsupervised?)
2. Determine data types
3. Augment data set with all the necessary columns (at this level, test and training sets should be all together or should you provide the logic to convert any new input into the appropriate transformed data? Maybe it’s the latter. Imagine you run a PCA to define the input space. You will not want to mix the test data to generate the PCA. It should be completely independent)
4. Filter data columns (this should also be embedded in pre-processing logic. Note, however, that for you to do data exploration you need the intelligibility of structured data. Suppose for example you bucket stuff, you need to keep the name of the bucket and the values! I.e. bucket = ‘Age below 14’ and value = 1 or 0 is not a good representation for visualization, you should have a column with Age\_bucket whose values are the names of the buckets and then plot those. Do maybe for data exploration you need to operate in the space of the augmented data set)



Data types



The support can be continuous or discrete and the type can be numeric or text. We could also add time-stamp types as a row.

Discrete variables will always come in pairs

* A text representation and
* A numeric representation of the level

Continuous variables will be bucketed. The buckets themselves become categorical variables, which will then come in pairs.

So it is probably best to screening for support, first:

* If categorical then make sure you have 2 representations and that’s it
* If continuous you have to bucket
  + If numeric, bucketing is easy
  + If time stamp also relatively easy if you can convert that into a temporary numerical representation and the buckets are just chinks of times
  + If text bucketing is hard and likely on a case-by-case basis

For data visualization, you might need to perform some importance analysis before, especially if the number of features is large.

Maybe better to split training and test set from get-go, so you get to a common ground where you have a test set and a training set. One distinguisher is also if training set is provided or not. If it’s not, you might have to shuffle the data first and then split it.

Visualization

Here it also depends on the type of problem. For supervised you can relate inputs to outputs. For unsupervised it is all about inputs’ relationship to one another.

That said it is probably useful to do some form of unsupervised stuff in supervised problems. Suppose you discover that there are clusters in the input features and they do not correspond with labeling. That is probably telling you something about the problem, e.g. classification is particularly difficult because you are trying to distinguish between elements within a cluster instead of distinguishing between clusters.

It seems one more reason to run some unsupervised before supervised.

Cross-validation comes AFTER you even analyzed the data